Database Processing CS 451 / 551

Lecture 12: Two-Phase Locking

Suyash Gupta
Assistant Professor
Distopia Labs and ONRG
Dept. of Computer Science
(E) suyash@uoregon.edu
(W) gupta-suyash.github.io

UNIVERSITY OF OREGON

Assignment 3 is Out!
Deadline: Nov 30, 2025 at 11:59pm

Final Exam: Dec 8, 2025 at 8-10am

How to Guarantee Serializability?

- If we know the full schedule (all the transactions that are part of the schedule) ahead of time, we can try to create a serializable schedule.
- Unfortunately, this is not a practical expectation.
- How to guarantee serializability?

Locks

- Use **Locks** to restrict access to database records.
- **Lock Manager →** Stores and grants access to Locks.

Locks

- Use **Locks** to restrict access to database records.
- **Lock Manager →** Stores and grants access to Locks.

Type	• **Shared Lock →** A shared lock on a data-item **D permits** concurrent access to the data-item **D** by multiple transactions. **Good for Reads**!
	• **Exclusive Lock →** An exclusive lock on a data-item **D disallows** concurrent access to the data-item **D** by multiple transactions (only one transaction at a time). **Good for Writes**!
Granularity	• Granularity defines the **level** at which a transaction acquires a lock. For example: a lock can be acquired for a full transaction or before access to a specific data-item.

Lock Compatibility Matrix

If a transaction Ti holds a S-Lock/X-Lock can another transaction acquire a S-Lock/X-Lock.

	Shared Lock (S-Lock)	Exclusive Lock (X-Lock)
Shared Lock (S-Lock)	✓	✗
Exclusive Lock (X-Lock)	✗	✗

Transaction Lock Phases

- First, each transaction **determines the type of lock** (S-Lock or X-Lock) it wants.

Transaction Lock Phases

- First, each transaction **determines the type of lock** (S-Lock or X-Lock) it wants.
- Next, it **requests the specific type lock for a data-item** from Lock Manager.

Transaction Lock Phases

- First, each transaction **determines the type of lock** (S-Lock or X-Lock) it wants.

- Next, it **requests the specific type lock for a data-item** from Lock Manager.

- **Two Possible Cases**:
  - **Transaction gets the requested lock for the data-item**

  - **Request Denied**

Transaction Lock Phases

- First, each transaction **determines the type of lock** (S-Lock or X-Lock) it wants.

- Next, it **requests the specific type lock for a data-item** from Lock Manager.

- **Two Possible Cases**:
  - **Transaction gets the requested lock for the data-item**
    - **Locks** the data-item.
    - Completes the desired task.
    - **Unlocks** the data-item and **releases** the lock back to Lock Manager.

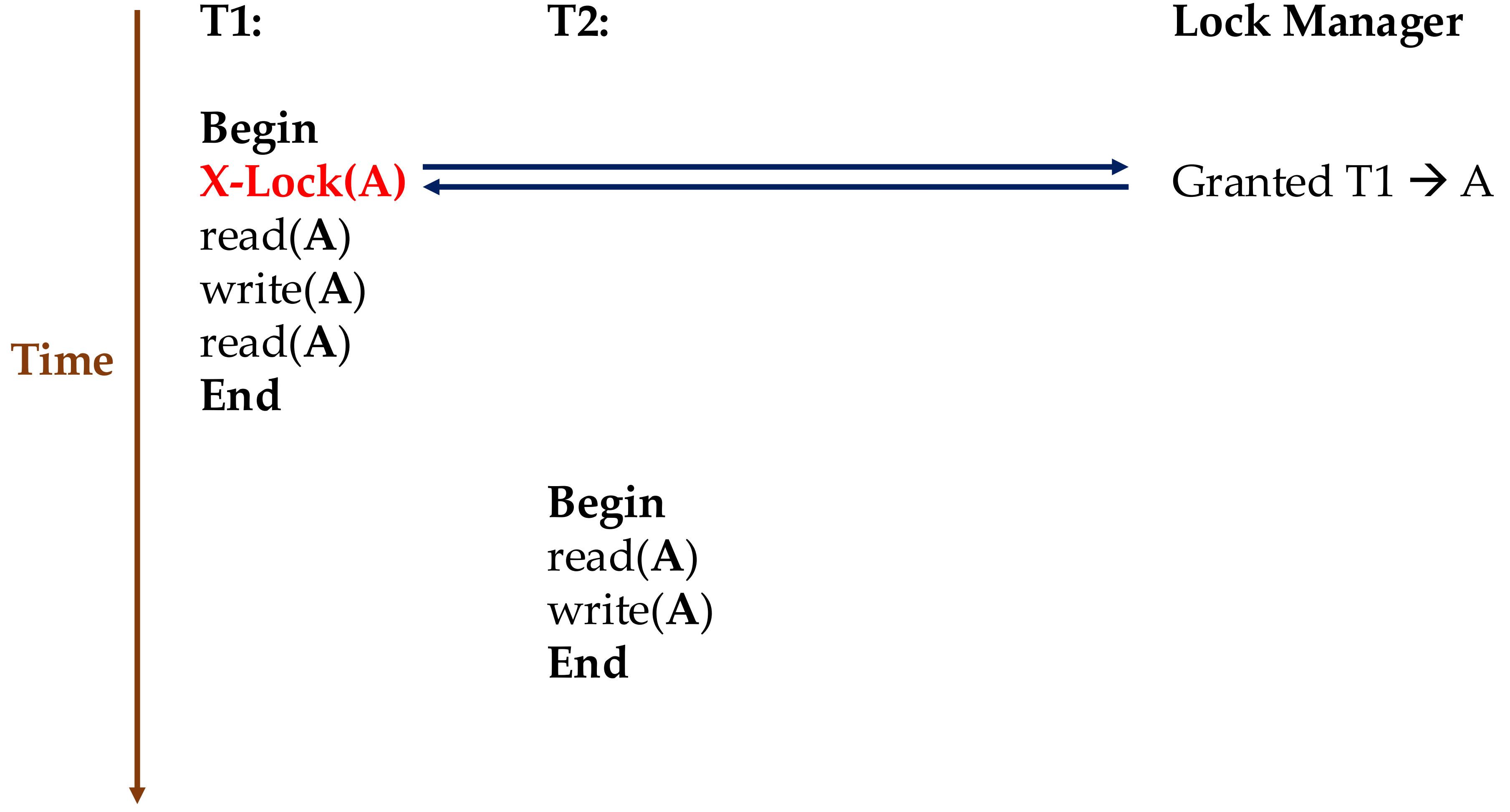
  - **Request Denied**

Locking Example I

Time

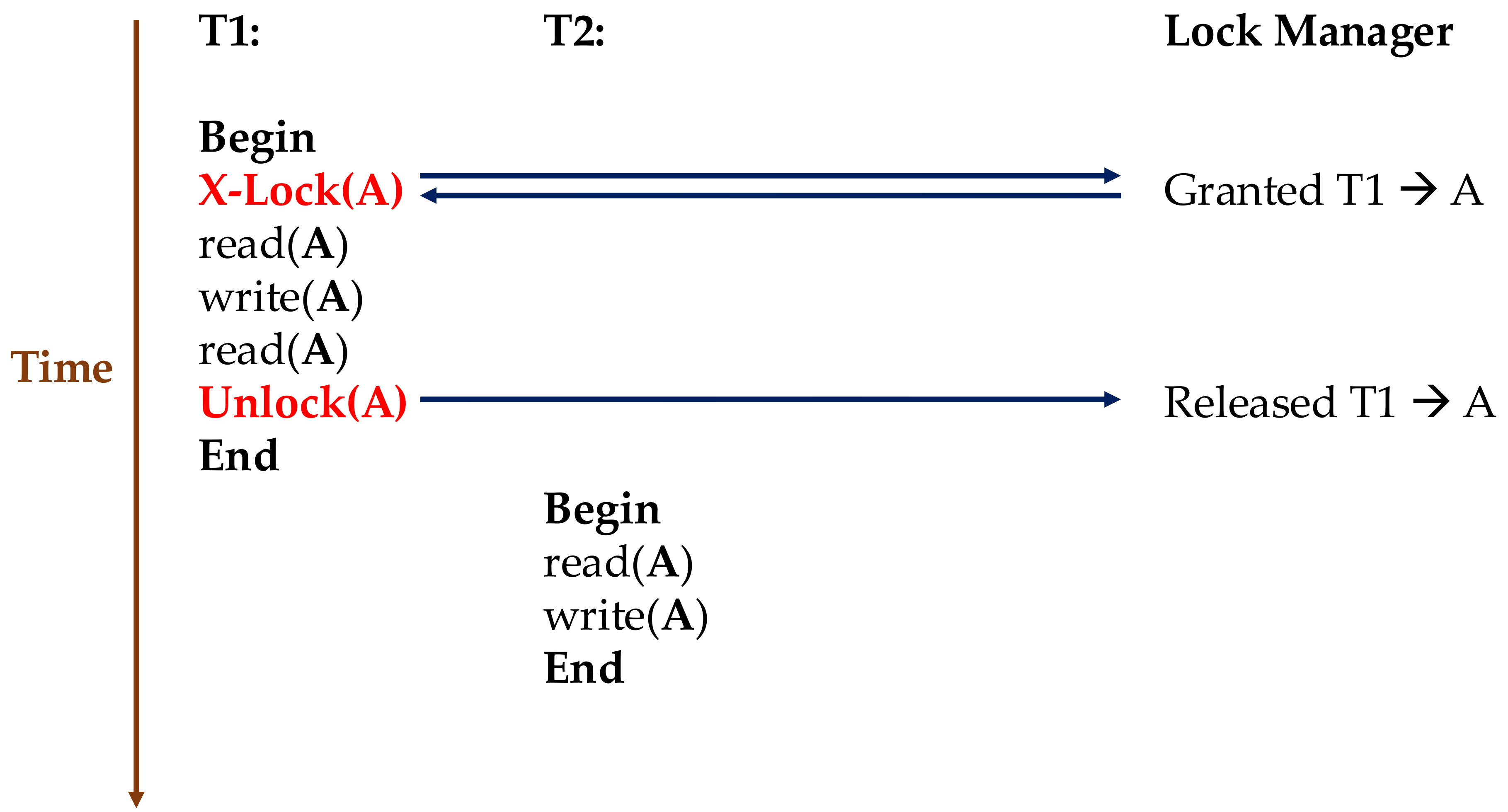
T1:	T2:	Lock Manager
Begin		
read(**A**)		
write(**A**)		
read(**A**)		
End		
	Begin	
	read(**A**)	
	write(**A**)	
	End	

Locking Example I

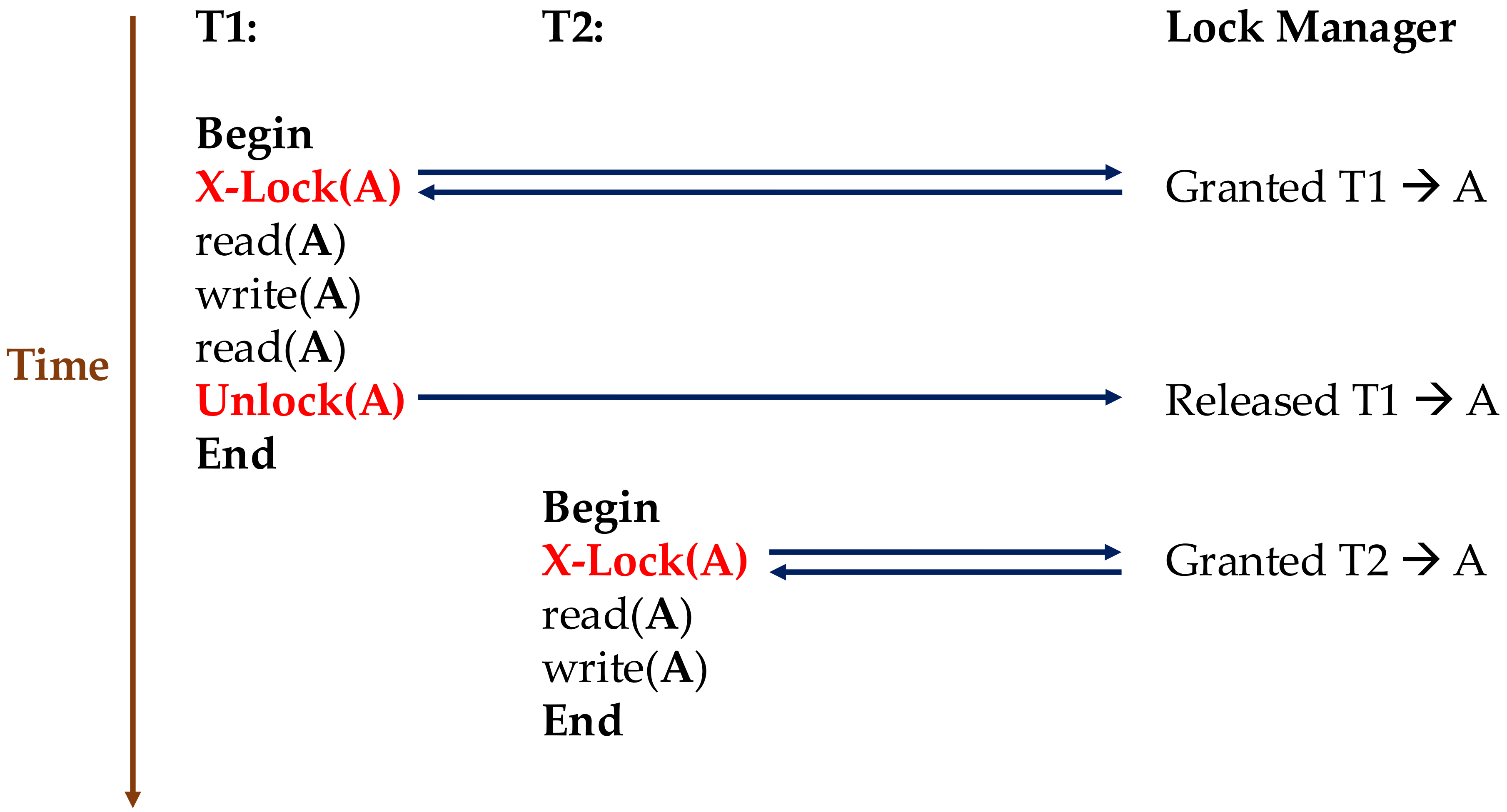


	T1:	T2:	Lock Manager
Time	Begin		
	X-Lock(A)		Granted T1 → A
	read(**A**)		
	write(**A**)		
	read(**A**)		
	End		
		Begin	
		read(**A**)	
		write(**A**)	
		End	

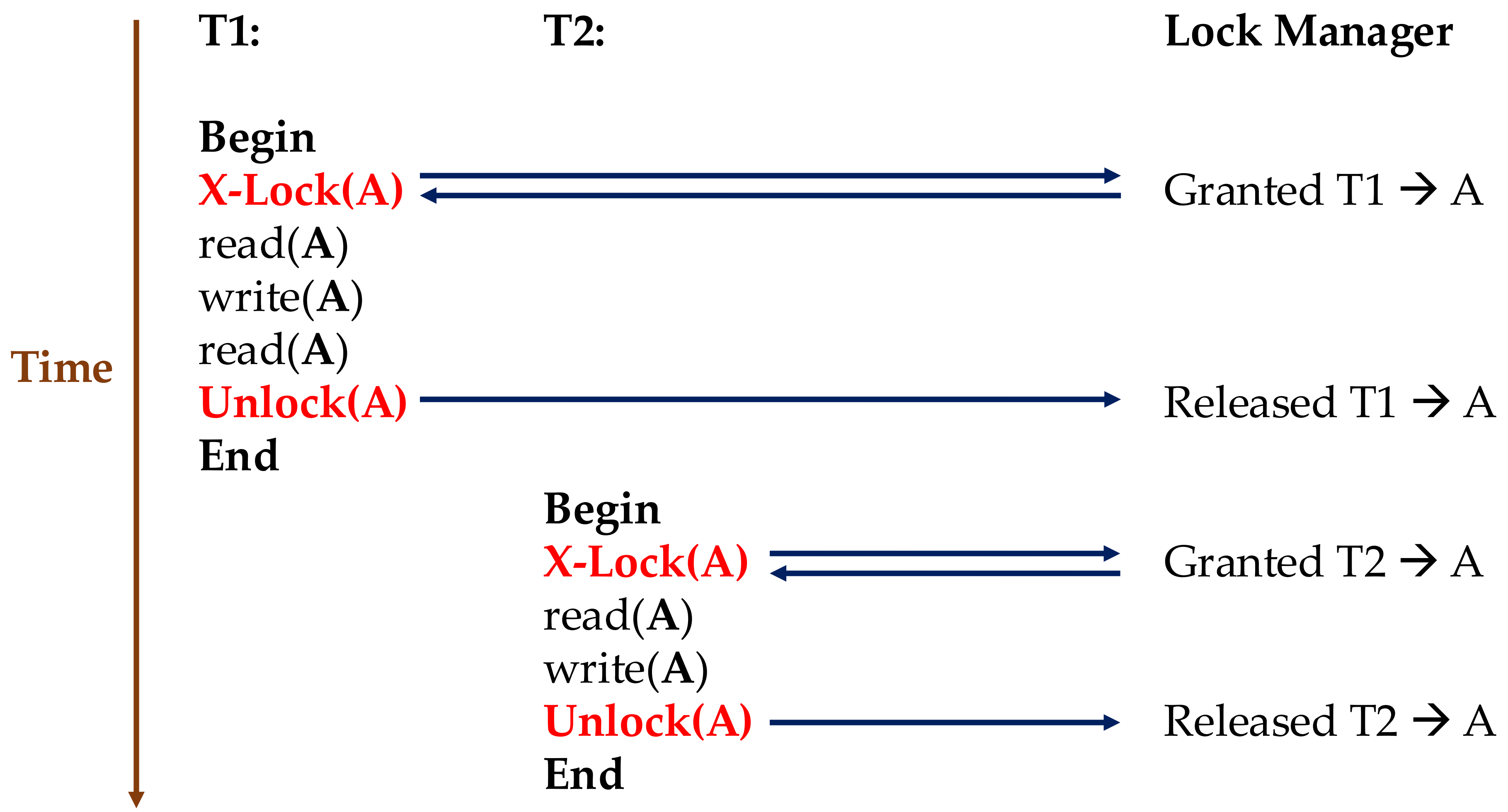
Locking Example I



Locking Example I



Locking Example I



Time	T1:	T2:	Lock Manager
	Begin		
	X-Lock(A)		Granted T1 → A
	read(**A**)		
	write(**A**)		
	read(**A**)		
	Unlock(A)		Released T1 → A
	End		
		Begin	
		X-Lock(A)	Granted T2 → A
		read(**A**)	
		write(**A**)	
		Unlock(A)	Released T2 → A
		End	

Locking Example II

Time

T1:	T2:	Lock Manager
Begin		
read(**A**)		
write(**A**)		
	Begin	
	read(**A**)	
	write(**A**)	
	End	
read(**A**)		
End		

Locking Example II

Time

T1:	**T2:**	**Lock Manager**
Begin		
X-Lock(A)		Granted T1 → A
read(**A**)		
write(**A**)		
	Begin	
	read(**A**)	
	write(**A**)	
	End	
read(**A**)		
End		

Locking Example II

Time

T1:	**T2:**	**Lock Manager**
Begin		
X-Lock(A)		Granted T1 $\rightarrow$ A
read(**A**)		
write(**A**)		
Unlock(A)		Released T1 $\rightarrow$ A
	Begin	
	read(**A**)	
	write(**A**)	
	End	
read(**A**)		
End		

Locking Example II

Time

T1:	T2:	Lock Manager
Begin		
X-Lock(A)		Granted T1 → A
read(**A**)		
write(**A**)		
Unlock(A)		Released T1 → A
	Begin	
	X-Lock(A)	Granted T2 → A
	read(**A**)	
	write(**A**)	
	End	
read(**A**)		
End		

Locking Example II

Time

T1:	**T2:**	**Lock Manager**
Begin		
X-Lock(A)		Granted T1 → A
read(**A**)		
write(**A**)		
Unlock(A)		Released T1 → A
	Begin	
	X-Lock(A)	Granted T2 → A
	read(**A**)	
	write(**A**)	
	Unlock(A)	Released T2 → A
	End	
read(**A**)		
End		

Locking Example II

Time

T1:	T2:	Lock Manager
Begin		
X-Lock(A)		Granted T1 → A
read(**A**)		
write(**A**)		
Unlock(A)		Released T1 → A
	Begin	
	X-Lock(A)	Granted T2 → A
	read(**A**)	
	write(**A**)	
	Unlock(A)	Released T2 → A
	End	
S-Lock(A)		Granted T1 → A
read(**A**)		
End		

Locking Example II

Time

T1:	**T2:**	**Lock Manager**
Begin		
X-Lock(A)		Granted T1 → A
read(**A**)		
write(**A**)		
Unlock(A)		Released T1 → A
	Begin	
	X-Lock(A)	Granted T2 → A
	read(**A**)	
	write(**A**)	
	Unlock(A)	Released T2 → A
	End	
S-Lock(A)		Granted T1 → A
read(**A**)		
Unlock(A)		Released T1 → A
End		

Locking Example II

Time

T1:	T2:	Lock Manager
Begin		
X-Lock(A)		Granted T1 → A
read(**A**)		
write(**A**)		
Unlock(A)		Released T1 → A
	Begin	
	X-Lock(A)	Granted T2 → A
	read(**A**)	
	write(**A**)	
	Unlock(A)	Released T2 → A
	End	
S-Lock(A)		Granted T1 → A
read(**A**)		
Unlock(A)		Released T1 → A
End		

Is this serializable?

Did locking help?

Locking Example II

Time

T1:	T2:	Lock Manager
Begin		
X-Lock(A)		Granted T1 → A
read(**A**)		
write(**A**)		
Unlock(A)		Released T1 → A
	Begin	
	X-Lock(A)	Granted T2 → A
	read(**A**)	
	write(**A**)	
	Unlock(A)	Released T2 → A
	End	
S-Lock(A)		Granted T1 → A
read(**A**)		
Unlock(A)		Released T1 → A
End		

Is this serializable?

Did locking help?

No!

Concurrency Control protocol: Two-Phase Locking

Two-Phase Locking

- Two-phase locking (2PL) protocol determines whether a transaction can access an object in the database at runtime.

- The 2PL protocol does not need to know all the queries that a transaction will execute ahead of time.

Two-Phase Locking

- **Two phases of 2PL**.

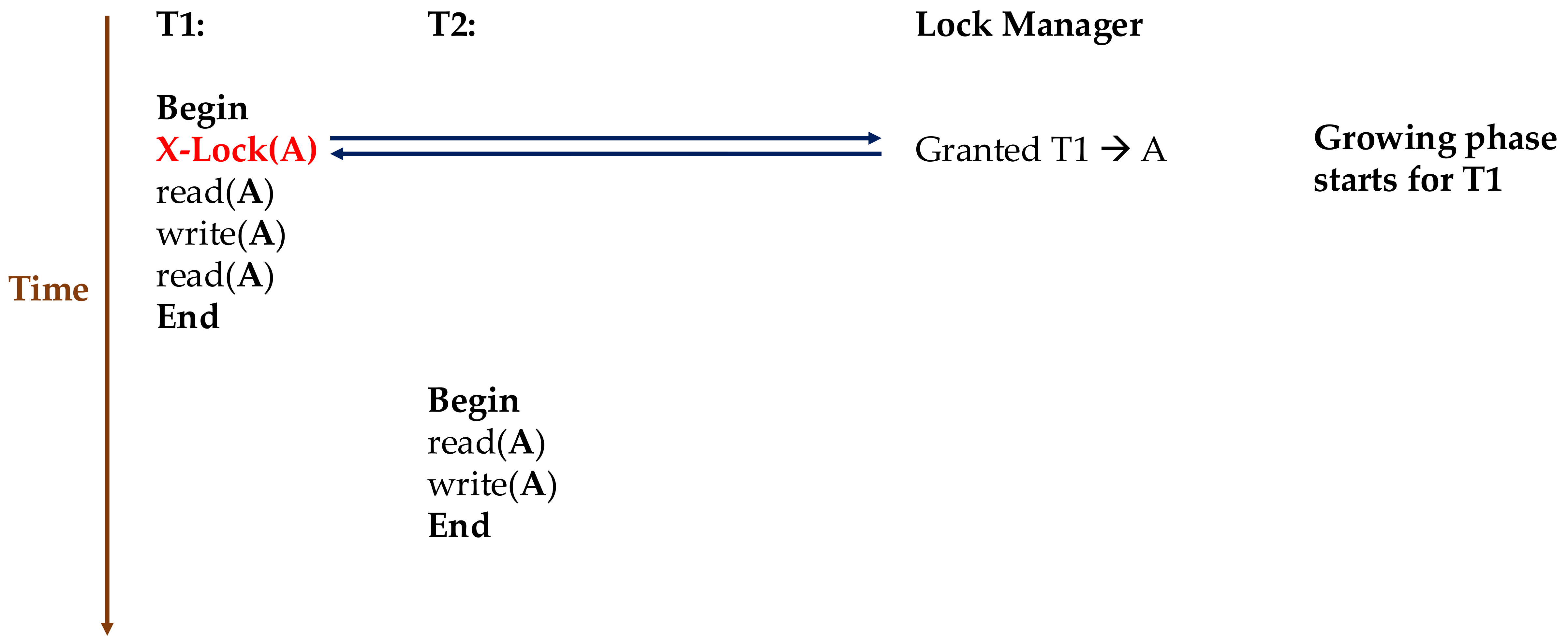
- **Growing Phase**:
  - Each transaction requests the locks that it needs from the Lock manager.
  - The lock manager grants/denies lock requests.

- **Shrinking Phase**:
  - A transaction is allowed to only release/downgrade locks that it previously acquired.
  - It cannot acquire new locks.
  - A transaction attempting to acquire a lock after releasing any lock is a violation!

2PL Example I

	T1:	**T2:**	**Lock Manager**
	Begin		
	read(**A**)		
	write(**A**)		
	read(**A**)		
Time	**End**		
		Begin	
		read(**A**)	
		write(**A**)	
		End	

2PL Example I



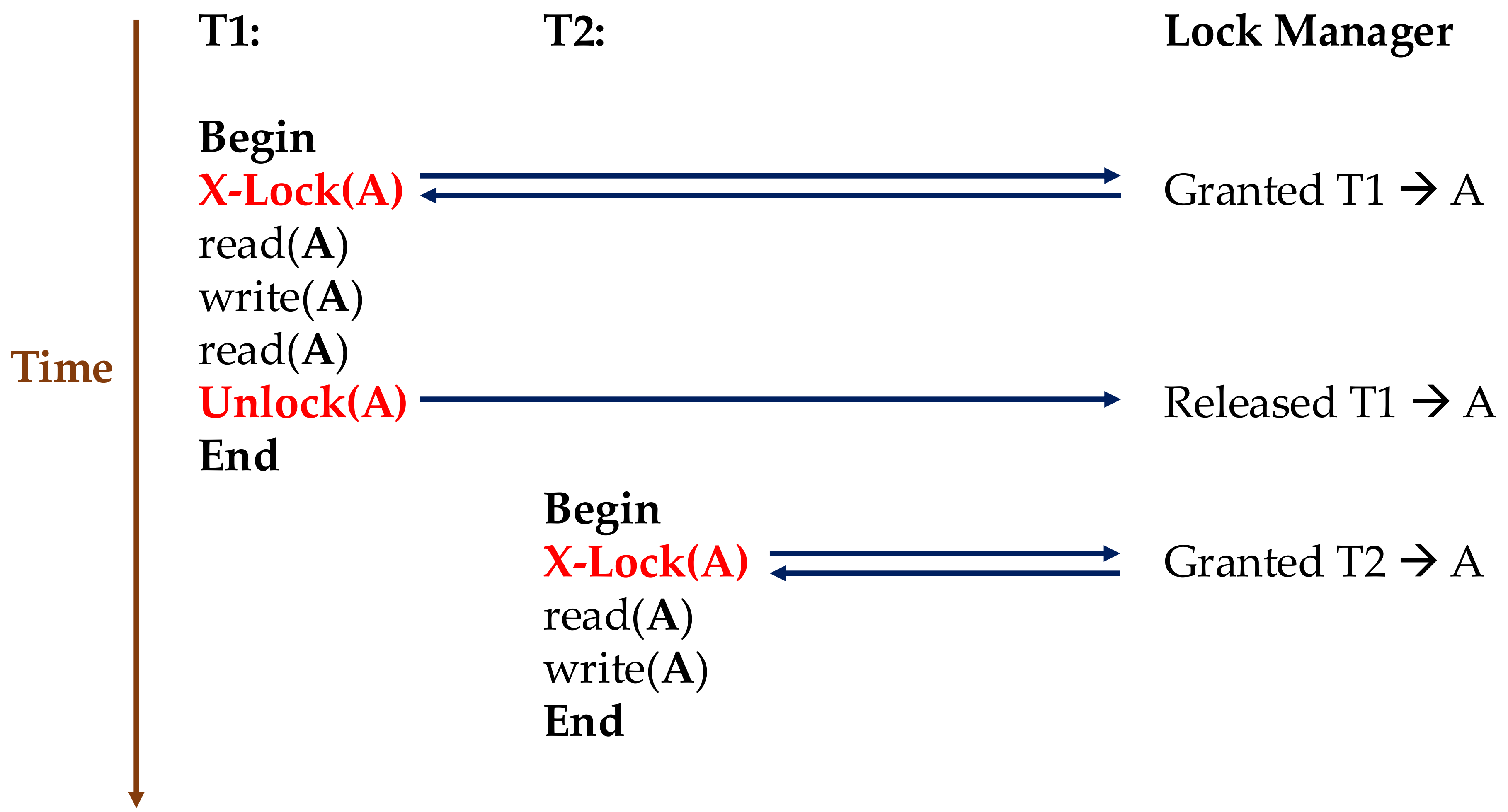
2PL Example I

Time

T1:	T2:	Lock Manager
Begin		
X-Lock(A)		Granted T1 → A
read(**A**)		
write(**A**)		
read(**A**)		
Unlock(A)		Released T1 → A
End		
	Begin	
	read(**A**)	
	write(**A**)	
	End	

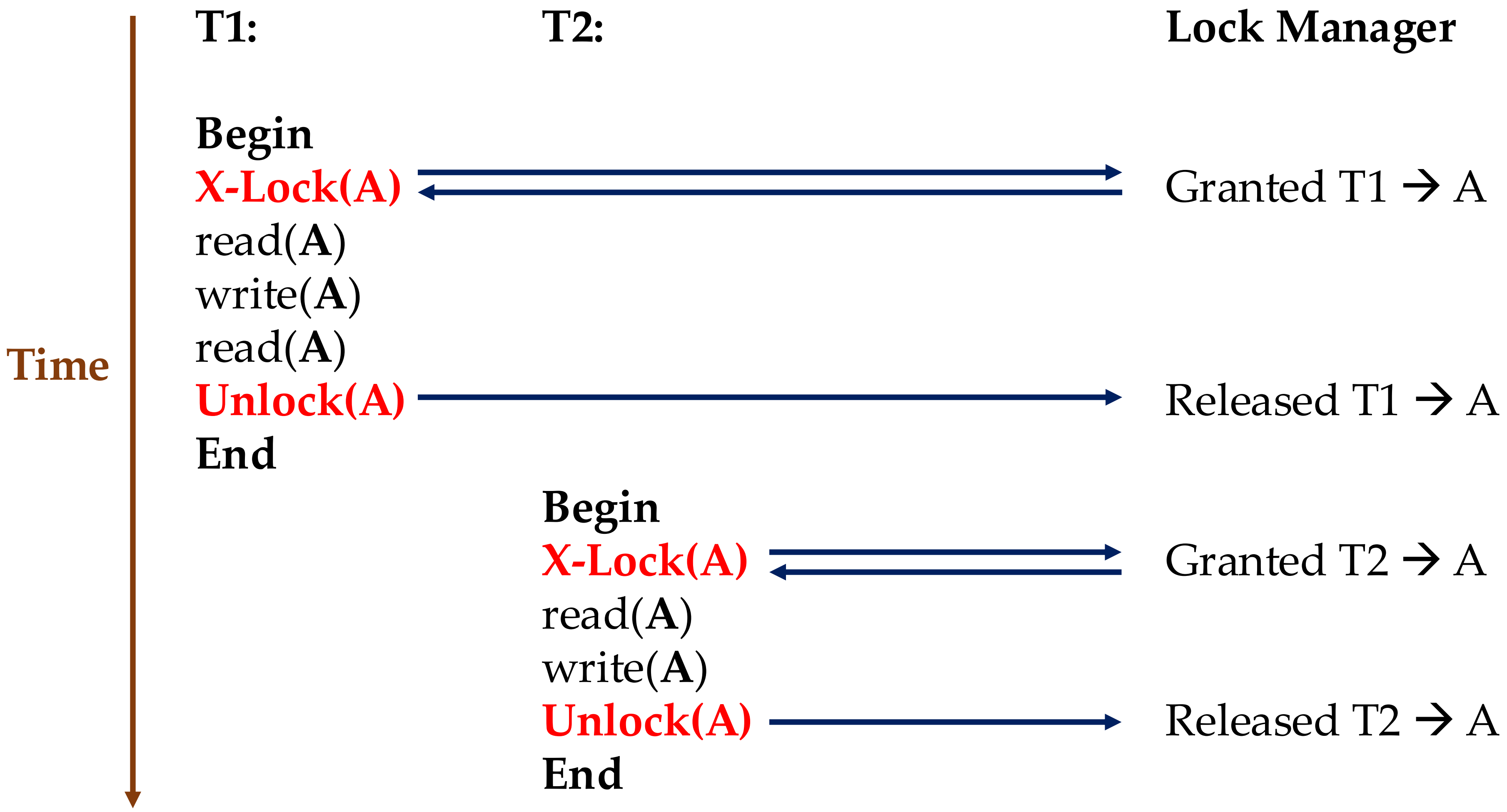
Growing phase ends and shrinking phase starts for T1.

2PL Example I



Growing phase starts for T2.

2PL Example I



Growing phase ends and shrinking phase starts for T2.

2PL Example I

Time

T1:	T2:		Lock Manager
Begin			
X-Lock(A)		$\rightleftarrows$	Granted T1 → A
read(**A**)			
write(**A**)			
read(**A**)			
Unlock(A)		$\rightarrow$	Released T1 → A
End			
	Begin		
	X-Lock(A)	$\rightleftarrows$	Granted T2 → A
	read(**A**)		
	write(**A**)		
	Unlock(A)	$\rightarrow$	Released T2 → A
	End		

Thus, this schedule follows 2PL.

Hence, serializable.

2PL Example II

Time	T1:	T2:	Lock Manager
	Begin		
	read(**A**)		
	write(**A**)		
		Begin	
		read(**A**)	
		write(**A**)	
		End	
	read(**A**)		
	End		

Let's try to force 2PL on this schedule.

2PL Example II

Time	T1:	T2:	Lock Manager	
	Begin			
	X-Lock(A)		Granted T1 → A	**Growing phase starts for T1**
	read(A)			
	write(A)			
		Begin		
		read(A)		
		write(A)		
		End		
	read(A)			
	End			

2PL Example II

Time	T1:	T2:	Lock Manager	
	Begin			
	X-Lock(A)		Granted T1 → A	Growing phase starts for T1
	read(A)			
	write(A)			
		Begin		
		X-Lock(A)	Request denied	
	read(A)			
	End			
		read(A)		
		write(A)		
		End		

2PL Example II

Time

T1:	**T2:**	**Lock Manager**	
Begin			
X-Lock(A)		Granted T1 → A	**Shrinking phase starts for T1**
read(**A**)			
write(**A**)			
	Begin		
	X-Lock(A)	Request denied	
read(**A**)			
Unlock(A)		Released T1 → A	
End			
	read(**A**)		
	write(**A**)		
	End		

2PL Example II

Time

T1:	**T2:**	**Lock Manager**	
Begin			
X-Lock(A)		Granted T1 → A	
read(**A**)			
write(**A**)			
	Begin		
	X-Lock(A)	Request denied T2	
read(**A**)			
Unlock(A)		Released T1 → A	**Growing phase starts for T2**
End			
	X-Lock(A)	Granted T2 → A	
	read(**A**)		
	write(**A**)		
	End		

2PL Example II

Time

T1:	T2:	Lock Manager	
Begin			
X-Lock(A)		Granted T1 → A	
read(**A**)			
write(**A**)			
	Begin		
	X-Lock(A)	Request denied T2	
read(**A**)			
Unlock(A)		Released T1 → A	**Shrinking phase starts for T2**
End			
	X-Lock(A)	Granted T2 → A	
	read(**A**)		
	write(**A**)		
	Unlock(A)	Released T2 → A	
	End		

Two-Phase Locking

- **2PL can guarantee conflict serializability** because it produces schedules whose dependency graphs are acyclic.

- But, what are the major challenges with 2PL?

Two-Phase Locking

- **2PL can guarantee conflict serializability** because it produces schedules whose dependency graphs are acyclic.

- But, what are the major challenges with 2PL?

- **Cascade aborts →** Aborting one transaction causes aborting all dependent transactions.

- **Deadlocks →** Two transactions waiting on resources held by each other.

Strong Strict Two-Phase Locking

Prevents Cascade Aborts

Strong Strict Two-Phase Locking

- A transaction is only allowed to release locks after it has ended (i.e., committed or aborted).

- Stricter than standard 2PL.
  - Smaller subset of schedules than standard 2PL allowed.

- Advantages:
  - No cascade aborts.
  - Aborted transactions can simply be undone!

Example

- Assume, the following two transactions, and initially **A = B = 1000**.

T1:	**T2:**
Begin	Begin
A = **A** − 100;	Print **A + B**
B = **B** + 100;	Commit
Commit	

Non 2PL

Time

T1:	**T2:**
Begin	**Begin**
X-Lock(A)	
read(**A**)	**S-Lock(A)**
A = A – 100	
write(A)	
Unlock(A)	
	read(**A**)
	Unlock(A)
	S-Lock(B)
X-Lock(B)	read(**B**)
	Unlock(B)
read(**B**)	Print **A+B**
B = B – 100	**Commit**
write(**B**)	
Unlock(B)	
End	

This is a serializable schedule but non-2PL!

2PL

Time

T1:	**T2:**
Begin	**Begin**
X-Lock(A)	
read(**A**)	**S-Lock(A)**
A = A – 100	
write(A)	
X-Lock(B)	
Unlock(A)	read(**A**)
read(**B**)	**S-Lock(B)**
B = B – 100	
write(**B**)	
Unlock(B)	
End	read(**B**)
	Unlock(A)
	Unlock(B)
	Print **A+B**
	Commit

This is a 2PL schedule and it is conflict serializable!

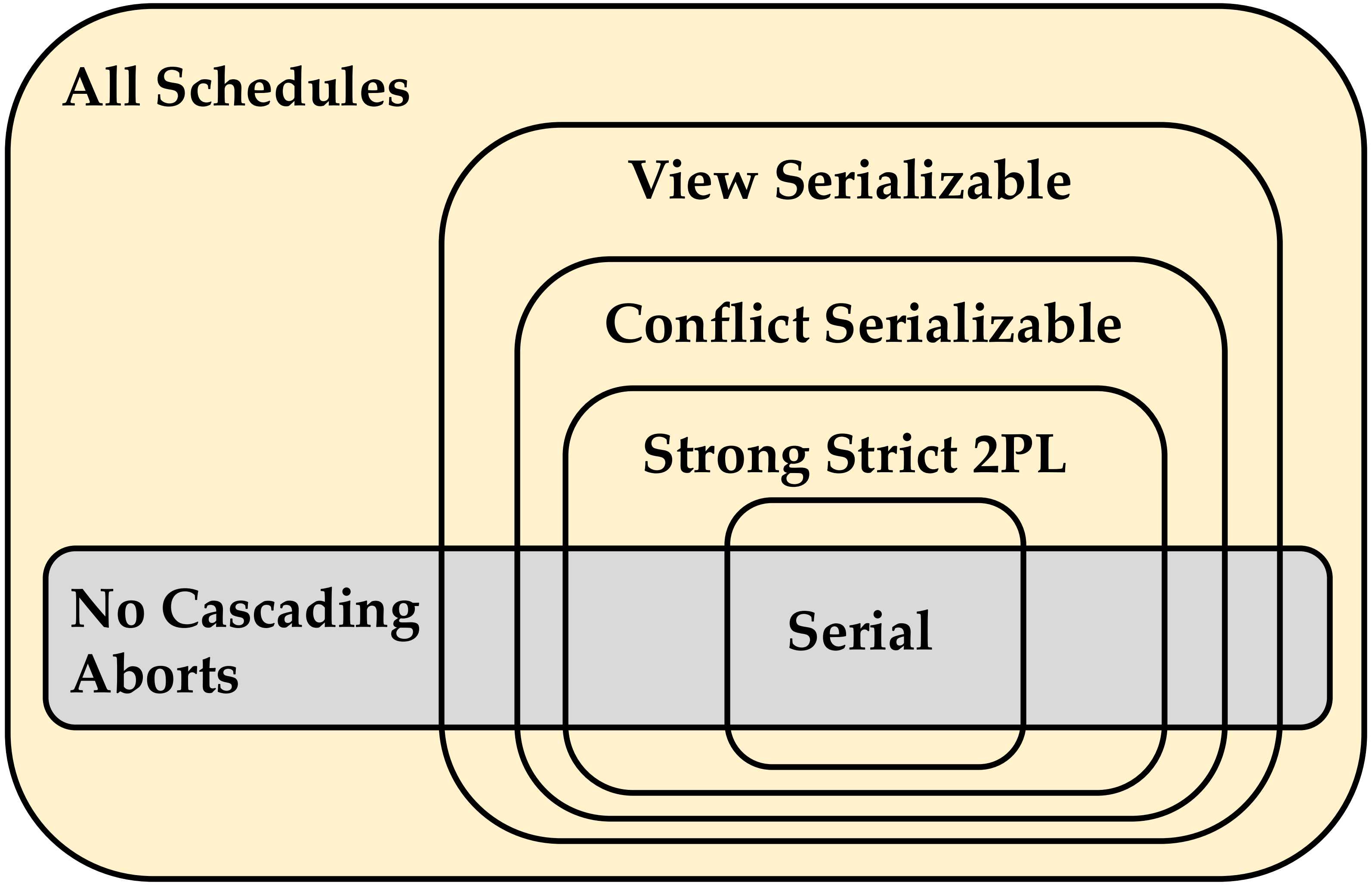
Strong Strict 2PL

Time

T1:	**T2:**
Begin	**Begin**
X-Lock(A)	
read(**A**)	**S-Lock(A)**
A = A – 100	↓
write(A)	↓
X-Lock(B)	↓
read(**B**)	↓
B = B – 100	↓
write(**B**)	↓
Unlock(A)	read(**A**)
Unlock(B)	**S-Lock(B)**
End	read(**B**)
	Print **A+B**
	Unlock(A)
	Unlock(B)
	Commit

This is a Strong Strict 2PL schedule and it will not suffer cascade aborts!

Strong Strict 2PL



Deadlocks in 2PL

Example

- Assume, the following two concurrent transactions.

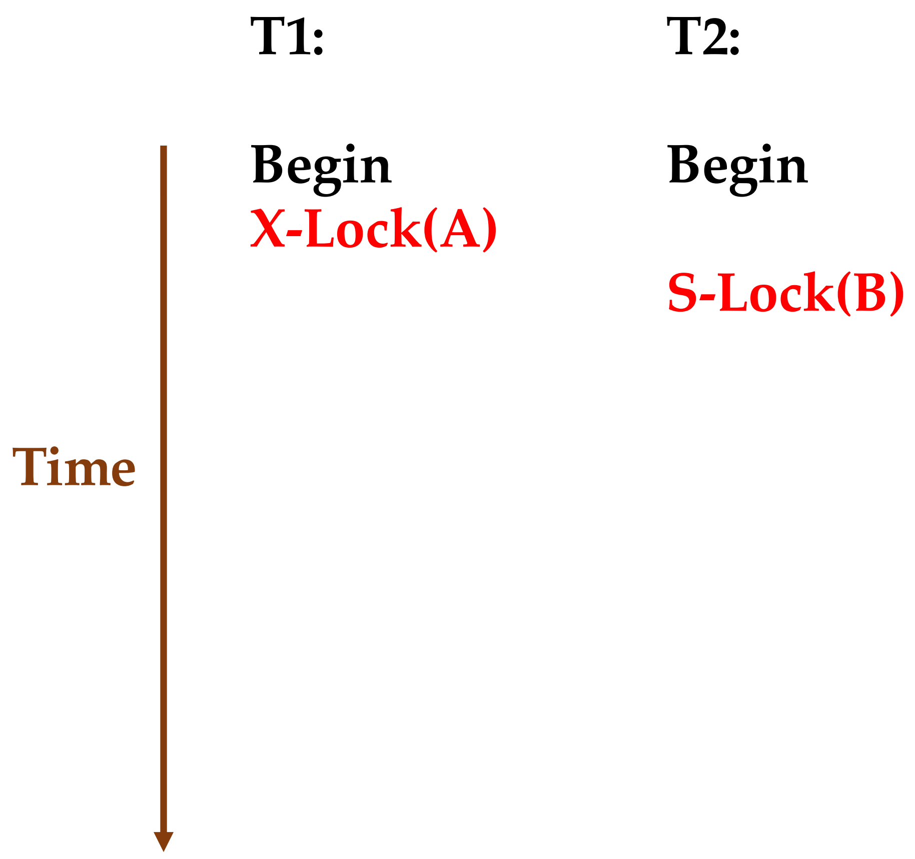
T1:	**T2:**
Begin	Begin
A = **A** − 100;	Print **B+A**
B = **B** + 100;	Commit
Commit	

Deadlock Example

Time

T1:	T2:
Begin	Begin
X-Lock(A)	

Deadlock Example



Deadlock Example

Time

T1:	T2:
Begin	**Begin**
X-Lock(A)	
	S-Lock(B)
read(**A**)	
A = A – 100	read(**B**)
write(A)	**S-Lock(A)**

T2 needs a lock on A.

Deadlock Example

Time

T1:	T2:
Begin	**Begin**
X-Lock(A)	
	S-Lock(B)
read(**A**)	
A = A – 100	read(**B**)
write(A)	**S-Lock(A)**
X-Lock(B)	

T1 needs a lock on B.

Both T2 and T1 are waiting for each other to release lock on other items.

Deadlock Management

- There are two ways to manage deadlocks:
- **Deadlock Detection** → When deadlock occurs, detect and solve.
- **Deadlock Prevention** → Prevent deadlock from occurring in the first place.

Deadlock Detection

Deadlock Detection

- Create a **waits-for graph**.
- Waits-for graph keep track of what locks each transaction is waiting to acquire.

Deadlock Detection

- Create a **waits-for graph**.

- Waits-for graph keep track of what locks each transaction is waiting to acquire.

- In the wait-for graph:
  - **Nodes** are transactions
  - **Add an Edge** from transaction **Ti** to **Tj** if **Ti** is waiting for Tj to release a lock.
  - The system periodically **checks for cycles** in waits- for graph and then decides **how to break it.**

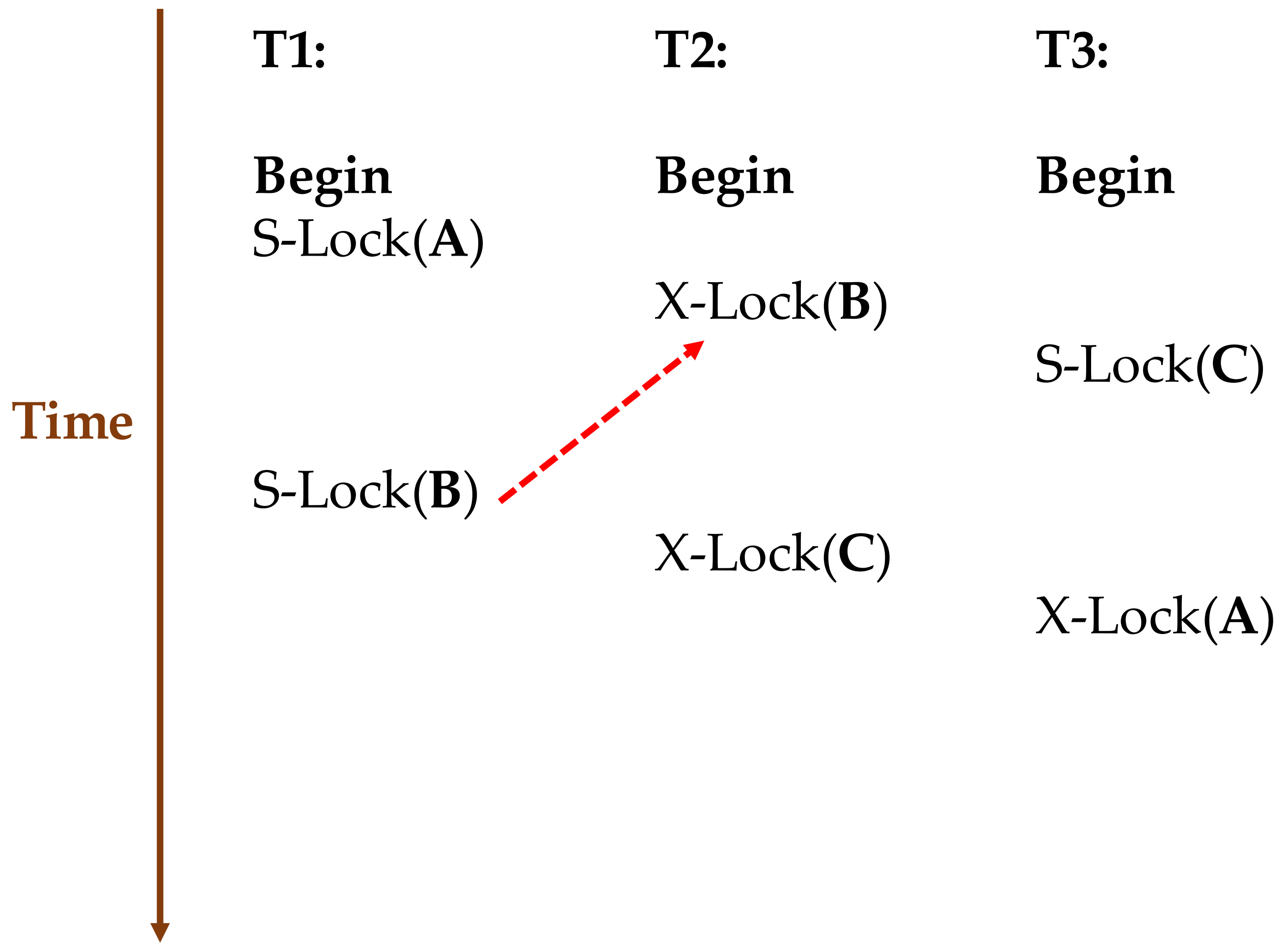
Deadlock Detection

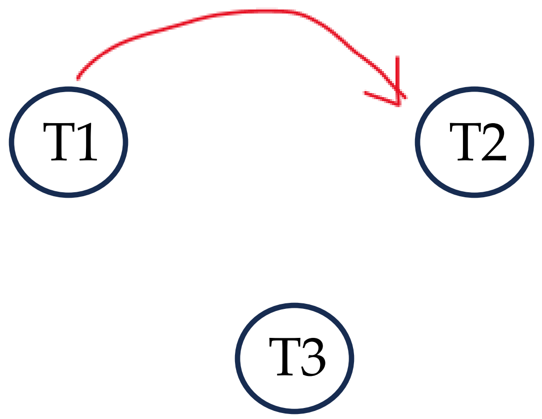
Time

T1:	T2:	T3:
Begin	**Begin**	**Begin**
S-Lock(**A**)		
	X-Lock(**B**)	
		S-Lock(**C**)
S-Lock(**B**)		
	X-Lock(**C**)	
		X-Lock(**A**)

Three transactions and three data-items

Deadlock Detection

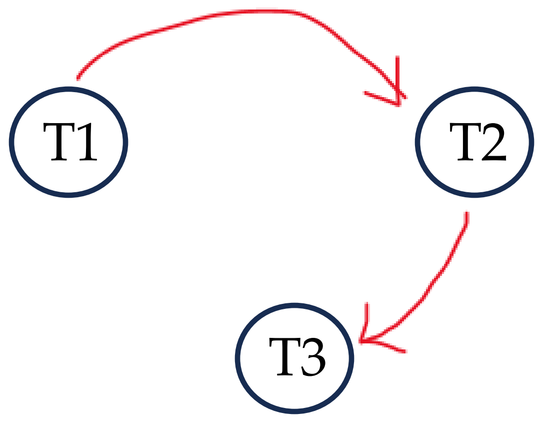




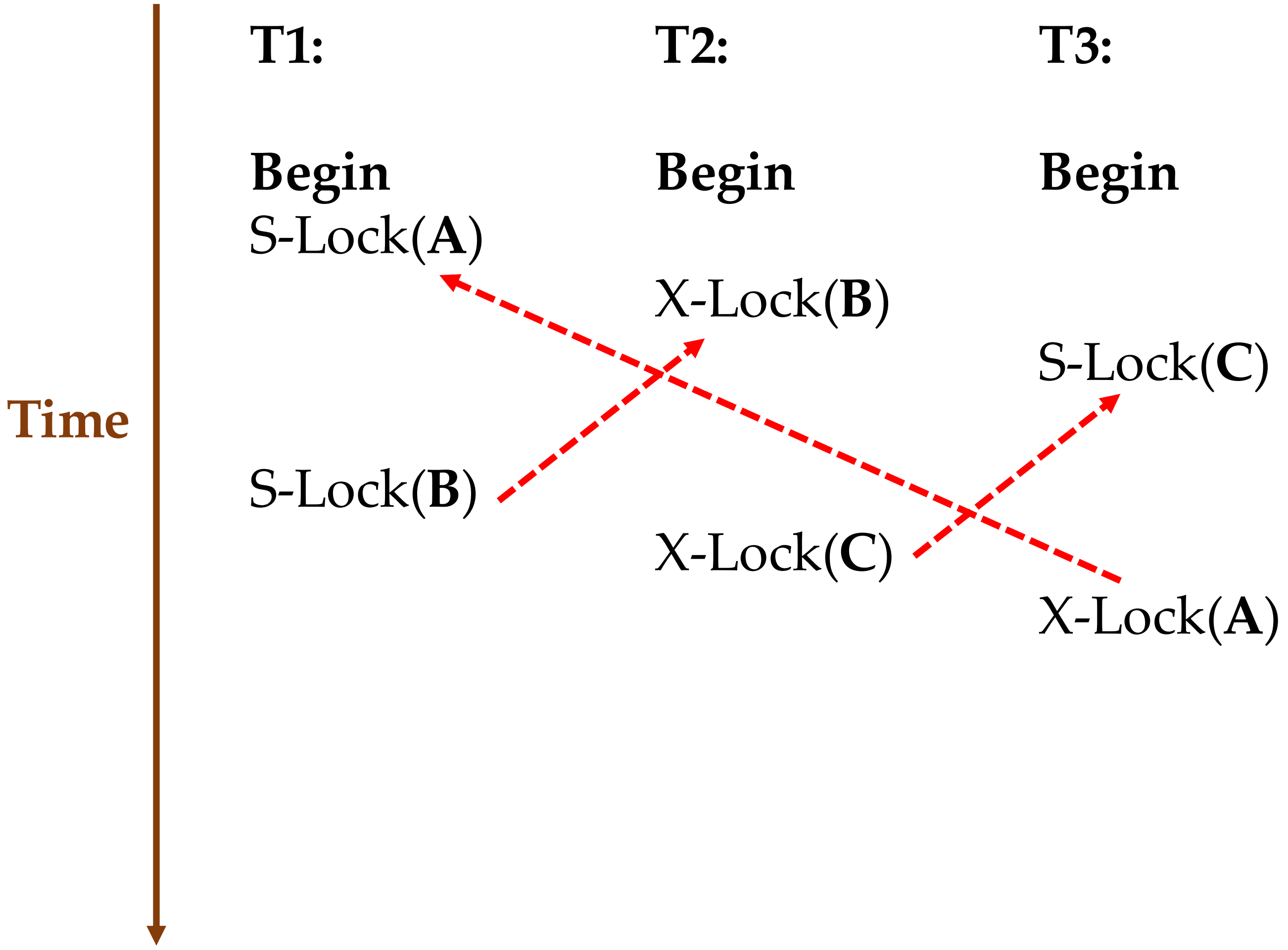
Deadlock Detection

Time

T1:	**T2:**	**T3:**
Begin	**Begin**	**Begin**
S-Lock(**A**)		
	X-Lock(**B**)	
		S-Lock(**C**)
S-Lock(**B**)		
	X-Lock(**C**)	
		X-Lock(**A**)

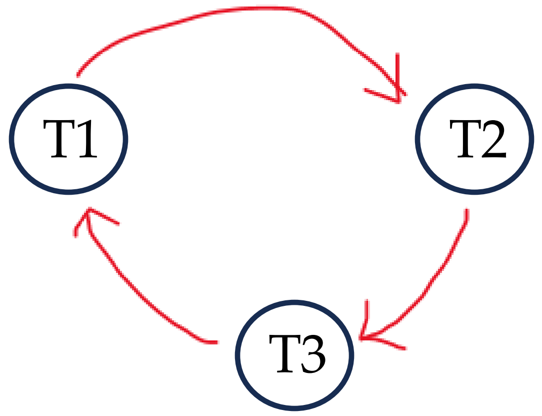


Deadlock Detection



Time

T1:	**T2:**	**T3:**
Begin	**Begin**	**Begin**
S-Lock(**A**)		
	X-Lock(**B**)	
		S-Lock(**C**)
S-Lock(**B**)		
	X-Lock(**C**)	
		X-Lock(**A**)



Deadlock Handling